# Variation-Aware Near Threshold Circuit Synthesis

Mohammad Saber Golanbari Saman Kiamehr Mojtaba Ebrahimi Mehdi B. Tahoori
Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
e-mails: {golanbari, kiamehr, mojtaba.ebrahimi, mehdi.tahoori}@kit.edu

***Abstract*—Near-Threshold Computing (NTC) is shown to be a promising approach for improving the energy efficiency of VLSI circuits. Nevertheless, by reducing the supply voltage the delay impact of process variation significantly increases, leading to up to 20× performance variation compared to the nominal voltage. As a result, it is wasteful of energy and performance to deal with such variation by increasing the timing margins, which is common in nominal voltage. Therefore, considering the impact of process variation during the near-threshold circuit design phase is of decisive importance. In this paper, we propose a *variation-aware synthesis flow* for NTC to address this problem. The objective is to improve the performance and energy efficiency of a circuit during design time by considering statistical variation information. This is done by providing variation information to the synthesis tool, evaluating the performance of the synthesized circuit by Statistical Static Timing Analysis (SSTA), and adjusting the timing constraints accordingly in an iterative manner. Simulation results for a set of benchmark circuits show that our proposed flow reduces the variation by 86.6% and improves the performance and energy by 24.9% and 7.4%, respectively, at the expense of 4.8% area overhead.**

## I. Introduction

Although the transistor count per chip is growing according to the Moore's law, power budget and power density prohibit the full utilization of the chip [1]. Since the power consumption of a circuit is highly dependent on the supply voltage, aggressive voltage scaling to the near-threshold region (a.k.a. NTC) is considered as an efficient way of reducing the power consumption [2], [3]. NTC provides a reasonable trade-off between the *energy efficiency* of a circuit, defined as the required energy to perform an operation, and its performance [4], [5]. As shown in Figure 1, the energy efficiency of NTC can be 10× better than in the super-threshold region while the performance is still acceptable for many application domains such as the Internet of Things (IoT).

The benefits of NTC are not easily accessible without overcoming several challenges such as *increased functional failure rate* in the circuit elements, *large performance drop* compared to the super-threshold region, and *increased performance variation* [4]. The increased functional failure is normally addressed by incorporating alternative designs which are more suitable for ultra-low voltage [6]. The large performance reduction is the price designers pay in order to achieve a better energy efficiency. Nevertheless, in some applications targeted for NTC such as mobile devices, the performance of the circuit is also important to the end user [7]. Furthermore, as presented in [4], the performance variation in the near-threshold region is up to 20× larger than that of the super-threshold region. Therefore, the timing margin to deal with delay variation, as it is widely used for super-threshold circuits, is no longer efficient for the NTC circuits mainly due to the extent of the variations.

In order to deal with huge performance variations, statistical information regarding the variation of circuit elements has to be considered during the logic synthesis phase. In the library pruning technique proposed in [9], [10], cells which exhibit higher performance variation are eliminated from the standard cell library in order to improve the characteristics of the synthesized circuit. However, by applying such a technique, cells with higher energy efficiency such as minimum sized cells are eliminated from the library for the sake of reducing the performance variation (as proposed by [10]). Hence, the energy efficiency of a circuit is limited when synthesized with a pruned library. Gate-sizing is another technique to improve the energy efficiency of the circuits in the near-threshold region [11]. However, the gate-sizing technique works on a fixed circuit topology after the logic synthesis and the technology mapping, and because of this limitation, its impact could be limited. In summary, a generic synthesis approach to address the huge variations in NTC is still missing.

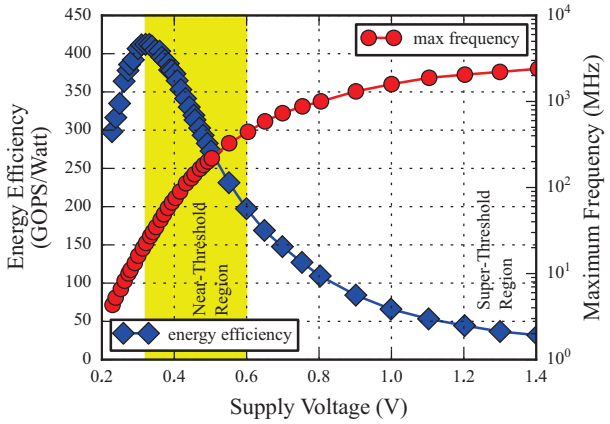


Fig. 1. Near-threshold operation is approximately 10× more energy efficient compared to super-threshold (65nm CMOS at 50°C) [8].

In this paper, we propose an *iterative variation-aware logic synthesis flow* to improve the performance and the energy efficiency of circuits in NTC, i.e. the goal is to optimize the statistical delay of the circuit considering the variations. This goal is achieved by performing SSTA to obtain the critical paths with most variation and applying appropriate measures based on the synthesis constraints to reduce the variation on those paths in an iterative manner. The proposed flow is wrapped around the standard (commercial) logic synthesis flow. Since the internal optimization engine of the synthesis tool is exploited, the proposed flow will automatically include all possible synthesis optimization techniques such as gate sizing, path restructuring, and logic borrowing. Optimization results show that this flow can reduce the delay variation by 86.6%, which translates into 24.9% better performance, and improve energy efficiency by 7.4%, at the expense of modest 4.8% area overhead. It is also possible to trade off energy efficiency and performance in this variation-aware flow.

The rest of the paper is organized as follows. In Section II, more details regarding NTC as well as related work are presented. Section III explains the proposed synthesis approach in details. In Section IV, the results of applying our approach to a set of benchmark circuits are presented, and in the end, Section V concludes the paper.

## II. Preliminaries

NTC comes with great promises and challenges. However, its widespread application is hampered by the intensified sensitivity to variations. In this section, we explain how variation affects the performance of the NTC circuits.

### A. Near-threshold computing

Aggressive supply voltage scaling has been known as an effective way to reduce the power consumption of circuits. Since

978-3-9815370-6-2/DATE16/©2016 EDAA

the dynamic (leakage) power has a quadratic (exponential) relation with the supply voltage, reducing the supply voltage decreases the overall power consumption of the chip. Based on this, extremely low-energy operation in the near-threshold region could be used to balance the performance and the energy efficiency of circuits. In this region, where the supply voltage is close to the transistor threshold voltage, the energy efficiency of circuits can be increased by 10× compared to super-threshold operation [4]. However, without appropriately addressing the performance variation in this region, the benefits of NTC is considerably reduced considering the extent of the variations.

### B. Variation in NTC

In the near-threshold region, the sensitivity of the circuit delay to temperature variation, supply voltage variation, and threshold voltage variation is significantly higher than in the super-threshold region [2]. In fact, decreasing the supply voltage by a few milli-volts pushes the circuit into the sub-threshold region, where the transistor current equation has an exponential relation with temperature, supply voltage, and threshold voltage. Therefore, any small fluctuation in the transistor threshold voltage has a large impact on the circuit delay. This intensifies the impact of different sources of process variation such as *Random Dopant Fluctuation* (RDF) and *Line Edge Roughness* (LER). As stated in [4], the performance variation in NTC can be 20× larger than in the super-threshold region. Without addressing such a substantial performance variation in the near-threshold region, exploiting the full benefits of NTC is not possible.

### C. Related work

The performance variation in NTC could be addressed by several techniques [8], [12]. Our proposed method lies within the logic synthesis category, and hence we review the techniques which deal with the performance variation by synthesis techniques.

*1) Library pruning:* The main idea of the library pruning is to remove the cells with high performance variation from the standard cell library. This reduces the overall variation of the cells in the library and eventually results in a smaller circuit performance variability. For example, it is suggested to remove the cells with high sensitivity to supply voltage variation [9]. In this technique, cells which exhibit a performance variation beyond a certain limit are excluded from the library. Another approach towards library pruning is to eliminate cells based on their size and structure [10], as the delay variation of a cell is correlated with its size and structure. As a result, small cells which exhibit higher variation are eliminated from the cell library. However, the eliminated small cells are beneficial, as they consume considerably less power compared to the larger cells. These cells could be used in shorter paths of a circuit to save the energy, without affecting the performance of the circuit determined by the timing of critical paths. Although library pruning can reduce the performance variation, the full capabilities of a circuit in terms of performance and energy efficiency cannot be exploited using this method.

*2) Gate-sizing:* Gate-sizing is a well-known technique to reduce the energy consumption of a circuit. A specific gate-sizing technique is proposed in [11], which employs As-Soon-As-Possible/As-Late-As-Possible (ASAP/ALAP) analysis to identify the cells which are not timing-critical. Although upsizing and downsizing of the gates would result in better energy efficiency, the improvement from this method is limited, as it does not incorporate other synthesis techniques, such as restructuring.

In the next section, we will explain our variation-aware circuit synthesis flow. This flow is wrapped around the existing commercial synthesis tools and leverages these tools in synthesizing circuits for NTC operation. Therefore, all synthesis techniques are automatically used to optimize the circuits.

## III. Proposed Approach

In this section, we begin by explaining how variation affects a circuit in NTC, and how our proposed approach will help to mitigate the impacts. For that, we first need to define some key terms. Figure 2 demonstrates the impact of performance variation on the delay of a path. The *nominal delay* of a path is the delay of that path without considering any variation. When considering a statistical variation, a path delay forms a statistical distribution. In this case, the path delay should be considered in such a way that the required yield is satisfied. For example $\mu+3\sigma$ of the delay distribution can be considered as the worst case which satisfies the yield constraints ($\mu$ and $\sigma$ are the mean and the standard deviation of the delay distribution, respectively). We refer to this value as the *corner delay* of the path. The difference between corner delay and nominal delay is considered as the *delay variation*.

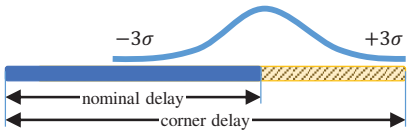


Fig. 2. Path delay w/o considering the impact of process variation i.e. nominal delay (blue), and considering the variation (hatched area).

### A. Motivational example

The amount of timing variation of a path is dependent on the number and the type of the cells in the path, and therefore, the timing variation can be considerably different from one path to another. Even if the circuit timing is initially balanced at the design time, this leads to unbalanced path delays considering variations (see Figure 3.a).

When no statistical information regarding the delay variation of the cells is provided to the synthesis tool, it tries to balance all path delays by loosening the shorter paths and tightening the critical paths according to their *nominal delays*. However, when the delay variation of a shorter path is very large, loosening such a path might lead to a larger circuit delay. In this case, since the synthesis tool is not aware of the *corner delay* of the paths (considering the delay variations), it cannot effectively improve the actual circuit timing, considering the delay variations. The purpose of our work is to inform the synthesis tool about the corner delay of the paths. By incorporating the accurate information regarding the path delays, the synthesis tool is able to save area and energy by loosening the short paths and spend the saved area and energy on the critical paths to improve the circuit performance. By doing so, the resulting timing of the synthesized circuit would look like Figure 3.b.

### B. Variation-aware synthesis flow

In this paper, we propose an iterative variation-aware synthesis flow to achieve this goal. Basically, two techniques are employed to inform the synthesis tool about the delay variation of the cells: First, a special library is used for synthesis, which contains the variation information of all the cells in the standard cell library (more details in Section III-B1). Second, SSTA is used to evaluate the performance of the synthesized circuits. Afterwards, the variation information from the SSTA is fed back to the synthesis tool by adjusting the timing constraints (more details in Section III-B2, and Section III-B3). In our approach, the synthesis tool is free to use any cell that operates correctly in NTC to improve the performance or the energy efficiency. However, by carefully adjusting the timing constraints and providing

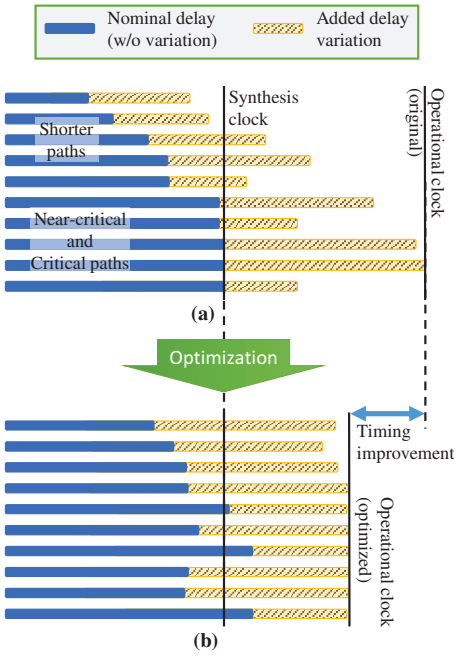


Fig. 3. **a)** Paths' delays of a circuit synthesized without variation information, and the impact of process variation on each path. **b)** Paths' delays of the same circuit after variation-aware synthesis.

variation information to the synthesis tool, the overall timing of the circuit (corner delay including variation) is improved.

A large circuit might have millions of paths. Therefore, it is not practical to apply constraints path by path. However, the paths end at circuit endpoints which are primary outputs, or flip-flop inputs. Fortunately, the number of endpoints is orders of magnitude smaller than the number of paths. Therefore, instead of setting constraints for specific paths, only the endpoints of the circuits are constrained. We can define the *nominal delay* and *corner delay* for each endpoint similar to the way they are defined for a path.

**Library characterization:** Before proceeding to the details of the iterative synthesis flow, it is necessary to have appropriate cell libraries for the synthesis. As shown in the library characterization part of Figure 4, based on the SPICE netlist and variation parameters such as $A_{vt}$ in Pelgrom's Model from [13], the cell library is characterized. The characterized *variation library* contains nominal as well as statistical information of the cell delays (statistical mean $\mu$ and standard deviation $\sigma$). This variation library is used by the SSTA tool to find the nominal and the statistical behaviors of the synthesized circuit. Later in Section III-B1, we will explain how the information from the variation library is used to create another library (*combined library*) which is solely used for the synthesis.

Figure 4 presents the proposed iterative synthesis flow. The goal of each iteration is to make the path delays more balanced considering the delay variations. In each iteration:

1) The circuit is synthesized with a combined library which contains variation information. The constraints of this synthesis are determined by the previous iteration (or no constraint for the first iteration).
2) The performance of the synthesized circuit under process variation is evaluated using SSTA. This gives the timing of all circuit endpoints.
3) Constraints are refined according to the extracted timing of each endpoint. The constraints of those endpoints which have a positive slack are relaxed while the constraints of endpoints with a negative slack are tightened.

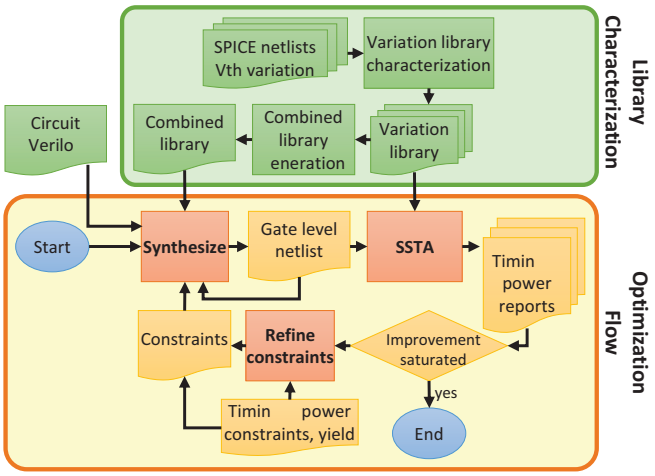


Fig. 4. Iterative variation-aware synthesis flow consists of two parts: 1) library characterization, 2) iterative optimization loop.

This loop continues until the circuit timing does not improve anymore. More details regarding each step are provided below.

*1) Synthesis with the combined library:* Although all variation information is available in the *variation library*, the synthesis tool is not able to use this information since it uses the nominal delay values as the metric. Therefore, the synthesis tool is not aware of the performance variation. It reads the delay and power information as well as the load capacitance of the cells from the library (without considering the corresponding variations). Based on this information, it decides which cell should be used and where. One way of providing the variation information to the synthesis tool is, by providing min-max libraries each containing minimum and maximum delays of each cell, respectively. This method is effective in the super-threshold region, where the amount of variation is small. In NTC, however, all cells have huge variations, and therefore, the result of min-max synthesis (and analysis) would give either too optimistic or too pessimistic results, which are not very helpful.

In order to provide the variation information to the synthesis tool in an effective way, information from the *variation library* is combined into another library named *combined library*. The trick is to use a combination of the mean and standard deviation of the delay values instead of the nominal delay values in order to guide the synthesis tool to choose more appropriate cells. Accordingly, we create the combined library by replacing each delay value in the cell library with $\mu + \beta\sigma$, where $\beta$ is a coefficient specifying the contribution of variation in the delays of the combined library, and $\mu$ and $\sigma$ are the mean and the standard deviation of the respective delay value from the *variation library*. In order to clarify, suppose that $\mu = 100ps$ and $\sigma = 20ps$ for a specific cell in the lookup table inside the *variation library*. If $\beta = 1$, the delay value which appears at the same location in the *combined library* would be $100ps + 1 \times 20ps = 120ps$. The combined library is just used for synthesis, as it contains information regarding variation. The reason is that, during synthesis, if the library contains no information regarding the variation of each cell (nominal library), the decisions of the synthesis tool are made based on the nominal cell timings. Therefore, the resulting performance variation of a circuit synthesized with such a library is not known. However, by providing the combined library

(which already contains the effect of the delay variation of each cell) to the synthesis tool, the synthesis tool can make more informed decisions on the choice of the cells and the resynthesis strategies.

*2) Performance evaluation with SSTA:* After each synthesis iteration, an SSTA is carried out. The SSTA uses the variation library for timing analysis and produces accurate variation-aware timing reports regarding the performance of the circuit. The statistical information from the SSTA will be used to adjust the timing constraints for the next iteration (Section III-B3). The *circuit delay* is the maximum of the corner delays of all the endpoints, and the slack of each endpoint is calculated as the difference between the corner delays of the circuit and the given endpoint.

*3) Timing constraints optimization:* In each iteration the target is to slightly improve the timing (i.e. reduce circuit corner delay to *target delay*). Based on the target delay, the constraints are updated for the next iteration, such that the timing and power are improved in the next iteration. These constraints are then fed back to the synthesis tool again. This loop continues until the improvement saturates i.e. no improvement is achieved in several subsequent iterations. Please note that, although spending more time on the synthesis phase might lead to a better result due to its heuristic nature, the limited time budget for finding the optimum result necessitates the termination of the optimization loop when the improvement saturates.

The method to refine the constraints for the circuit endpoints is demonstrated in Figure 5. Timing constraints are adjusted based on the SSTA timing reports for all endpoints. Suppose that $T$ is the *circuit corner delay* calculated using the SSTA. This is calculated as the maximum of all the endpoints' corner delays (considering the variations). We want to reduce $T$ by a factor of $\gamma$ in each iteration (improving the performance). Therefore, the *target delay* ($D$) for the next iteration would be:

$$D = (1 - \gamma)T. \tag{1}$$

Based on $D$ and the endpoint delays calculated by the SSTA, we can update the endpoint slacks for the next iteration:

$$s_{i,SSTA} = D - d_{i,SSTA} \tag{2}$$

where $d_{i,SSTA}$ is the delay of endpoint $i$ calculated by the SSTA, and $s_{i,SSTA}$ is the statistical slack of endpoint $i$, respectively. When in iteration $k$, the slack of an endpoint is negative ($d_{i,SSTA} > D$) as in Figure 5.a, the paths ending to that endpoint should be tightened. But if the slack is positive ($d_{i,SSTA} < D$) as in Figure 5.b, the paths ending to the endpoint should be relaxed. Therefore, the constraints of all the endpoints for iteration $k+1$ are tightened or relaxed by the corresponding amount of $s_{i,SSTA}$, respectively.

## IV. Experimental Results

In order to show the effectiveness of our proposed flow, we applied it to ITC'99 benchmark circuits. We have also compared it with the library pruning technique presented in [9].

### A. Simulation setup

The cells from Nangate 45nm Open Cell Library are characterized for different supply voltages, ranging from 0.45V to 1.1V with Cadence Virtuoso Variety statistical characterization tool. The threshold voltages of NMOS and PMOS transistors of Nangate library are 471mV and -423mV, respectively. Therefore, the specified supply voltage range covers both super-threshold and near-threshold regions.

The *variation library* is created in the Effective Current Source Model (ECSM) format using Cadence Virtuoso Variety and contains the statistical information regarding the cells. This library is then converted into the *combined library* as described in Section III. The

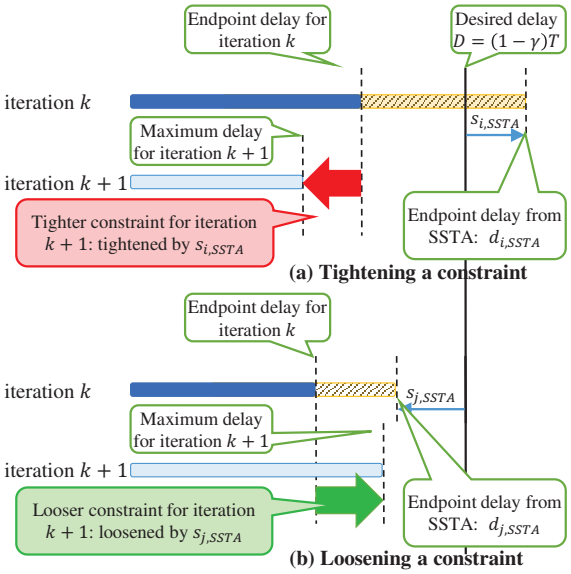


Fig. 5. **a)** According to the timing of endpoint $i$ calculated by the SSTA ($d_{i,SSTA}$) in iteration $k$, a tighter constraint is applied to this endpoint for iteration $k+1$, **b)** According to the timing of endpoint $j$ calculated by the SSTA ($d_{j,SSTA}$) in iteration $k$, a looser constraint is applied to this endpoint for iteration $k+1$.

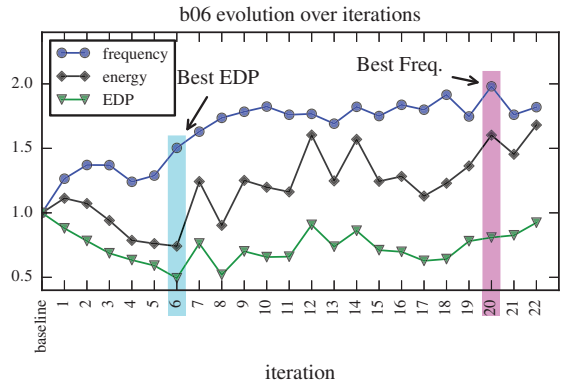


Fig. 6. b06 circuit evolution over iterations: from baseline to iteration 22 ($V_{dd} = 0.45$, $\gamma = 0.05$). All values are normalized to the baseline.

optimum $\beta$ coefficient for creating such a library is dependent on the circuit and the cell library. We tried a range of coefficients and empirically determined $\beta = 0.5$ to work the best for our synthesis. Synopsys Design Compiler is used for synthesis, and Cadence Encounter Timing System is used for SSTA. The *energy per cycle* is then calculated based on the reported leakage power, dynamic power, circuit nominal delay, and circuit corner delay (from SSTA).

### B. Detailed results for a single circuit

In this section, the detailed result for a single circuit is explained. Thereafter, the results for all circuits are presented in less detail for brevity. Figure 6 shows the evolution of b6 benchmark circuit over the optimization iterations. Frequency, energy (per clock cycle) and Energy-Delay-Product (EDP) are normalized with respect to the *baseline* in order to easily follow the improvement compared to the baseline. The *baseline* is the synthesized circuit using the traditional flow (with a normal library).

As shown in this figure, the maximum frequency of the circuit follows a rising trend over iterations, which means the performance

TABLE I. OPTIMIZATION RESULTS FOR ITC'99 BENCHMARK CIRCUITS($V_{dd} = 0.45$, $\gamma = 0.05$).

| benchmarks | | | | | | library pruning [9] | | | | proposed | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| circuit | gates | flip-flops | frequency (MHz) | area ($\mu m^2$) | energy per cycle (fJ) | variation reduct. (%) | frequency imprv. (%) | area imprv. (%) | energy imprv. (%) | optimum-EDP iter. | variation reduct. (%) | frequency imprv. (%) | area imprv. (%) | energy imprv. (%) | runtime overhead (×) |
| b01 | 65 | 5 | 358.68 | 61.71 | 0.31 | 27.3 | 10.5 | -9.5 | -7.0 | 7 | 36.6 | 14.5 | 2.2 | 12.2 | 5.0 |
| b02 | 32 | 4 | 444.84 | 28.20 | 0.10 | 5.3 | 2.1 | -15.1 | -3.8 | 2 | 36.0 | 16.6 | -1.9 | -3.2 | 1.1 |
| b03 | 185 | 30 | 331.24 | 193.12 | 1.03 | -7.1 | -2.3 | 12.0 | 11.4 | 0 | 0 | 0 | 0 | 0 | 0.8 |
| b04 | 1071 | 66 | 115.82 | 1070.92 | 11.50 | -16.1 | -3.8 | 15.0 | 16.6 | 2 | 14.2 | 3.6 | 8.9 | 11.0 | 2.0 |
| b05 | 2176 | 34 | 80.47 | 2107.25 | 26.70 | 23.8 | 5.6 | 5.3 | 13.8 | 15 | 106.2 | 31.4 | -3.3 | 10.3 | 7.4 |
| b06 | 74 | 9 | 273.60 | 74.48 | 0.38 | 63.4 | 28.1 | 9.3 | 12.9 | 6 | 96.8 | 50.3 | 10.7 | 25.8 | 4.0 |
| b07 | 560 | 44 | 174.03 | 577.22 | 3.72 | 27.7 | 8.8 | -3.7 | 1.8 | 7 | 55.9 | 19.6 | -9.5 | -0.4 | 4.0 |
| b08 | 204 | 21 | 253.10 | 219.98 | 1.28 | -29.3 | -7.7 | 11.5 | 10.5 | 2 | -14.3 | -3.9 | 8.22 | 6.02 | 1.0 |
| b09 | 224 | 28 | 243.90 | 239.40 | 1.59 | -33.8 | -10.1 | 22.4 | 22.7 | 9 | 56.3 | 23.2 | -22.0 | -11.2 | 6.5 |
| b10 | 227 | 17 | 222.22 | 223.71 | 1.24 | -11.6 | -3.4 | 11.2 | 8.5 | 7 | 82.4 | 32.8 | -31.5 | -14.1 | 2.8 |
| b11 | 865 | 30 | 152.91 | 929.67 | 7.17 | -11.4 | -3.4 | 10.3 | 15.7 | 2 | 29.1 | 10.0 | 0.9 | 7.5 | 2.7 |
| b12 | 1561 | 120 | 172.03 | 1523.91 | 9.30 | -25.2 | -7.0 | 15.5 | 19.9 | 0 | 0 | 0 | 0 | 0 | 0.5 |
| b13 | 364 | 48 | 267.88 | 376.39 | 1.75 | 24.9 | 9.3 | 8.1 | 9.6 | 2 | 17.2 | 6.2 | 8.5 | 13.1 | 2.0 |
| b14 | 12530 | 215 | 23.16 | 12681.02 | 488.33 | 67.4 | 13.3 | 2.2 | 14.6 | 9 | 143.7 | 33.3 | -9.1 | 8.5 | 3.3 |
| b15 | 9215 | 417 | 70.49 | 9287.66 | 107.84 | -36.4 | -7.7 | 5.8 | 4.1 | 12 | 67.1 | 18.2 | -5.3 | 5.1 | 3.5 |
| b17 | 26509 | 1317 | 63.26 | 27356.24 | 340.88 | -38.7 | -7.9 | 3.8 | -0.2 | 27 | 102.1 | 29.3 | -8.5 | 7.9 | 6.4 |
| b18 | 89049 | 3020 | 19.39 | 90167.08 | 3471.53 | -1.6 | -0.3 | 2.7 | 4.3 | 28 | 181.6 | 43.5 | -9.0 | 15.6 | 10.0 |
| b19 | 164552 | 6042 | 17.91 | 168536.00 | 6900.49 | 34.7 | 5.8 | 3.5 | 10.3 | 24 | 234.2 | 58.0 | -7.0 | 23.7 | 13.4 |
| b20 | 26226 | 430 | 20.72 | 26660.91 | 1179.30 | 41.1 | 7.1 | 3.8 | 12.7 | 19 | 206.7 | 50.4 | -11.2 | 14.6 | 6.3 |
| b21 | 25847 | 430 | 21.29 | 26330.81 | 1137.99 | 79.0 | 15.0 | 3.5 | 16.8 | 25 | 195.4 | 47.6 | -12.4 | 12.1 | 7.4 |
| b22 | 39451 | 613 | 22.01 | 39982.73 | 1667.53 | 5.5 | 0.9 | 3.9 | 8.8 | 12 | 171.0 | 37.5 | -8.9 | 10.4 | 4.4 |
| average | | | | | | 9.0 | 2.5 | 5.8 | 9.7 | 10.3 | 86.6 | 24.9 | -4.8 | 7.4 | 4.5 |

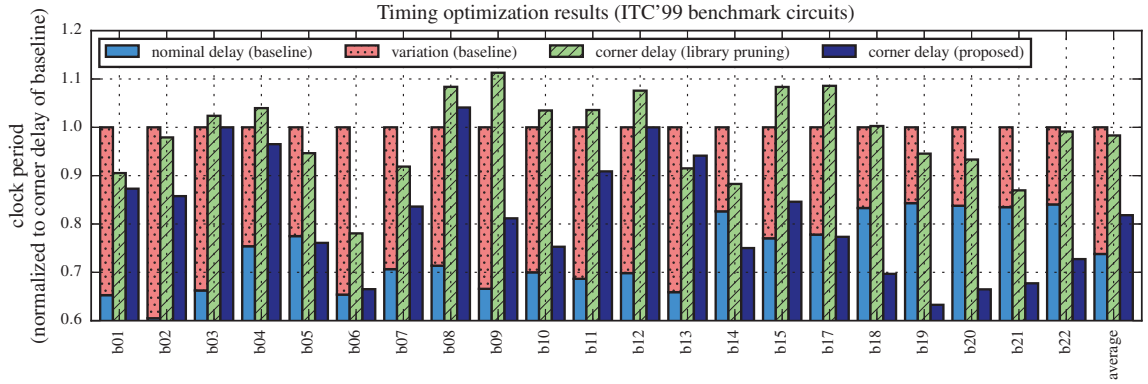


Fig. 7. *Clock period* comparison of ITC99 benchmark circuits with different methods ($V_{dd} = 0.45$, $\gamma = 0.05$). The contributions of nominal delay and variation for baseline are also presented.

of the circuit is improving compared to the baseline. On the other hand, the energy initially drops, but after some iterations it increases again. This means that by applying new constraints in each iteration, the synthesis tool is able to reduce the corner delay of the critical paths while retaining the power from shorter paths. However, after a certain iteration the synthesis tool has to increase the energy of the circuit in order to apply the new constraints. Therefore, there is an optimum point considering both energy and timing. This point can be discovered by calculating the EDP of the circuit. At the best EDP iteration shown in Figure 6, the maximum frequency is improved by 50.3% and the energy is reduced by 25.8%.

Another optimization target marked in this figure represents the iteration with the best performance i.e. best frequency. Although this leads to a 100% performance improvement, the energy overhead is also 50% which is not in line with the energy efficiency expected in NTC. For this circuit, we can conclude that iteration 6 is the optimal result for this circuit. In this paper, our target is to optimize both energy efficiency and performance i.e. optimize the EDP.

### C. Results for all benchmarks

The optimization results for all ITC'99 benchmark circuits are presented in Table I. We also implemented the library pruning method proposed in [9] for the sake of comparison. For each benchmark circuit, frequency, area, and energy per cycle of all circuits before optimization (i.e. baseline) and after optimization are extracted. For the optimized circuits, these values are presented as improvement percentages i.e. frequency (performance) improvement, area improvement, and energy improvement over the baseline synthesis. The presented variation improvement is calculated as:

$$\text{variation improvement (\%)} = \left(1 - \frac{\text{corner delay}_{\text{optimized}} - \text{nominal delay}_{\text{baseline}}}{\text{corner delay}_{\text{baseline}} - \text{nominal delay}_{\text{baseline}}}\right) \times 100 \quad (3)$$

For the presented simulation setup, the main improvement of the library pruning technique is the energy efficiency (9.7%), since it is not able to effectively improve the timing (just 2.5%). Our investigation reveals that by applying the library pruning technique, some fast cells are eliminated from the cell library, which prevents the synthesis tool from achieving better performance. In addition to that, when the library has fewer cells (as in the pruned library), the synthesis tool has fewer options for improvement. In contrast, our proposed method reduces the variation by 86.6% which corresponds to 24.9% better performance, as well as 7.4% better energy efficiency on average.

We have found that our proposed variation-aware synthesis flow is able to reduce the corner delay to be even smaller than the nominal delay achieved by the conventional synthesis flow for the largest benchmarks (i.e. b14, b17-b22), where the variation improvement

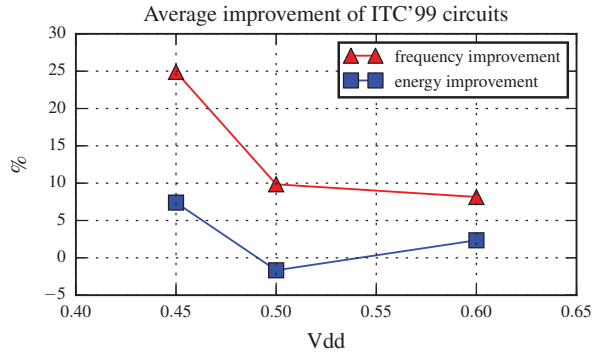


Fig. 8. Average improvement of ITC99 benchmark circuits over different supply voltages.

calculated by Equation (3) is more than 100%, as shown in Table I. The reason is that the synthesis tool puts a limited effort to optimize a circuit, and this effort is distributed over different endpoints to satisfy the constraints. In case the circuit is small, a considerable effort is put on each endpoint during the synthesis, which fiercely optimizes the circuit from the very first iteration. Therefore, for small circuits the synthesis flow usually saturates in a few iterations (b01-b13). For circuit b03 and b12, since no improvement is observed over iterations, the synthesis flow sticks to the baseline. The synthesis effort is initially distributed among all the endpoints which leads to a limited improvement when just one synthesis is performed (i.e. the baseline or the library pruning method). By defining constraints for the endpoints, we direct the synthesis tool to put more efforts on the critical endpoints (considering variations). As a result, the improvement from the iterative variation-aware synthesis is considerably large. Especially for large circuits which have a wide variety of short and critical paths, the improvement is comparatively higher. This highlights the scalability of the proposed approach and its effectiveness for large circuits. In summary, the energy efficiency of NTC is maintained while improving the performance.

The runtime overhead of the iterative variation-aware synthesis flow is on average 4.5× more than the conventional synthesis. Since the incremental synthesis feature of the synthesis tool is exploited, the runtime overhead is considerably smaller than the number of the required iterations to find the optimum EDP point. In each optimization iteration, the full state of the synthesis environment is loaded from the previous optimization iteration, and an incremental synthesis is performed with the new constraints.

As presented in Table I, the number of the iterations and consequently the runtime is higher for larger circuits. Because these circuits are large, there might be many opportunities for improving their performance or energy efficiency. As a result, the synthesis tool is able to continue the optimization process for many iterations. This can be fixed by considering a higher $\gamma$ coefficient e.g. 0.1, which can effectively decrease the number of iterations. However, since for the other circuits in Table I, the results are reported for $\gamma = 0.05$, for the sake of consistency, the results for large circuits are also reported for the same $\gamma$.

Figure 7 compares the timing improvement of all ITC'99 benchmark circuits. The corner delay of the library pruning and our proposed flow are also shown in this figure. As illustrated, the corner delays of b17-b21 circuits optimized with the proposed flow are even smaller than the nominal delay of the respective baselines (i.e. without considering the variations). Although for these circuits the flow imposes an area overhead, the performance and energy improvements are also significant (see Table I).

### D. Improvement of the proposed method with $V_{dd}$ reduction

Figure 8 presents the average of the improvements (performance and energy) of the benchmark circuits for different supply voltages. The average performance improvement is maximized at $V_{dd} = 0.45$, where the amount of variation is higher comparatively. The energy improvement is also better for the same supply voltage. When the amount of variation is larger, the difference between the corner delays of the endpoints is also larger. Thus, by improving a smaller number of critical endpoints (in terms of timing), a large performance improvement can be achieved. At the same time, the energy efficiency can be improved by loosening the shorter paths without affecting the performance of the circuit. However, when the amount of variation is smaller, the synthesis flow needs to deal with many critical or near-critical endpoints. This means that the synthesis tool needs to distribute the optimization effort among many paths, which usually leads to a smaller improvement. Therefore, we can conclude that on average the improvement of our method increases as the supply voltage reduces.

## V. Conclusion

NTC is a promising solution to address energy challenges of modern VLSI circuits. Although NTC leads to better energy efficiency compared to the super-threshold operation, the increased performance variation reduces the overall efficiency. In this work, we presented a variation-aware synthesis flow for NTC circuits to improve the performance of circuits operating in the near threshold region. The results show that the proposed method can reduce the variation by 86.6% (which means 24.9% better performance), and additionally improve energy efficiency by 7.4% with a small area overhead (4.8%).

## References

[1] B. Zhai *et al.*, "Theoretical and practical limits of dynamic voltage scaling," in *Design Automation Conference*, 2004, pp. 868–873.

[2] D. Marković *et al.*, "Ultralow-power design in near-threshold region," *Proceedings of the IEEE*, vol. 98, no. 2, pp. 237–252, 2010.

[3] B. H. Calhoun and A. Chandrakasan, "Characterizing and modeling minimum energy operation for subthreshold circuits," in *International Symposium on Low Power Electronics and Design*, 2004, pp. 90–95.

[4] R. Dreslinski *et al.*, "Near-Threshold Computing: Reclaiming Moore's Law Through Energy Efficient Integrated Circuits," *Proceedings of the IEEE*, vol. 98, no. 2, pp. 253–266, Feb 2010.

[5] S. Borkar and A. A. Chien, "The future of microprocessors," *Communications of the ACM*, vol. 54, no. 5, pp. 67–77, 2011.

[6] L. Chang *et al.*, "An 8T-SRAM for variability tolerance and low-voltage operation in high-performance caches," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 4, pp. 956–963, 2008.

[7] M. Severson, K. Yuen, and Y. Du, "Not so fast my friend: Is near-threshold computing the answer for power reduction of wireless devices?" in *Design Automation Conference*, 2012, pp. 1160–1162.

[8] H. Kaul *et al.*, "Near-threshold voltage (NTV) design: opportunities and challenges," in *Design Automation Conference*, 2012, pp. 1153–1158.

[9] J. Crop *et al.*, "Design automation methodology for improving the variability of synthesized digital circuits operating in the sub/near-threshold regime," in *International Green Computing Conference and Workshops*, 2011, pp. 1–6.

[10] S. Jain *et al.*, "A 280mV-to-1.2V wide-operating-range IA-32 processor in 32nm CMOS," in *IEEE International Solid-State Circuits Conference*, 2012, pp. 66–68.

[11] N. Conos *et al.*, "Maximizing yield in Near-Threshold Computing under the presence of process variation," in *International Workshop on Power and Timing Modeling, Optimization and Simulation*, 2013, pp. 1–8.

[12] M. R. Kakoee *et al.*, "Automatic synthesis of near-threshold circuits with fine-grained performance tunability," in *International Symposium on Low-Power Electronics and Design*, 2010, pp. 401–406.

[13] K. Kuhn *et al.*, "Process Technology Variation," *IEEE Transactions on Electron Devices*, vol. 58, no. 8, pp. 2197–2208, Aug 2011.